*“Be careful about reading health books. You may die of a misprint.*”Mark Twain

|  |  |
| --- | --- |
| *Chapter heading:* Advanced Topics  *Concepts to cover*   * Go-Live * Monitoring & maintaining * Updates to data-type rules |  |

Over the last few chapters we have gained an understanding of the HealthVault application programming interface (Chapter 3), learned about building a HealthVault web application with a focus on the HealthVault data types (Chapter 4), and built an engaging mobile application (Chapter 5). An application’s life cycle typically involves testing the application, releasing it to the user, and then monitoring it for anomalies, tasks that entail special requirements in a HealthVault context. This chapter will highlight best practices for releasing, maintaining, and marketing HealthVault application to end users.

# Testing your application

Well written software goes through multiple test cycles, including both automated and manual tests. This section outlines some valuable test scenarios around HealthVault account management, API interfaces, and data types, which you should consider in addition to other tests.

HealthVault enables people to share and manage multiple health records. It’s important to make sure that your application will work with multiple records associated with the same HealthVault account. In Chapter 3, we covered account management and ways to configure record switching. You need to ensure your application works with an account with multiple records. The best way to achieve this is to create several test accounts with multiple health records and try your application with them.

Another important aspect of account management is sharing. You can test this by sharing a healthvault record with another person and then making sure they can authenticate your application in association with that record. In case of insufficient permissions, your application should show an error message.

HealthVault provides an XML-based Web API. This API is accessible through programming libraries available through various HealthVault libraries and SDKs, as discussed in Chapter 3. While developing an application, you should pay special attention to any failure codes returned from HealthVault. In fact, make sure you log non-success return codes from API calls to HealthVault so that you can investigate reasons for failure. HealthVault provides a comprehensive list of status codes returned by the service at <http://msdn.microsoft.com/en-us/library/hh567902.aspx>. Particularly interesting is CREDENTIAL\_TOKEN\_EXPIRED, which your application should handle by requesting a new credentials token from HealthVault. The HealthVault .NET SDK and Mobile SDKs handle this status appropriately for you. In case you see the INVALID\_XML status code, you should look closer at your request to make sure the XML is valid for various HealthVault methods and data type schemas, which are available at http://developer.healthvault.com.

HealthVault enables a coherent and well adopted data ecosystem. It is very important to make sure that your application works well in this data ecosystem and uses the HealthVault data types appropriately. Chapter 4 explains HealthVault data types in detail. The best way to that make sure you are reading a data type appropriately is to create a few instances of the type you are consuming with various possible permutations and combinations and then access them in your application. Get Real Consulting offers a great tool, HealthVault X-ray, <http://xray.getrealconsulting.com/>, which enables you to create myriad instances of the data type you are consuming in an appropriate test account.

The second aspect of working with the data types is to make sure that the information you are writing is consumable by other applications. The HealthVault team offers a tool called HealthVault Data type checkup at http://datacheckup.healthvault-ppe.com/hvappcheckup, it works against data written by your application in a test record and finds any compatibility issues. Currently, the HealthVault data checkup tool supports only a limited number of types. Another mechanism to ensure that you play well is to copy your test records through HealthVault X-Ray in a production test account, authorize other HealthVault applications to access this account, and then confirm that the information is consumable by these applications.

Frequently, applications code properties of HealthVault data types improperly, or are not able to parse a flexible date format as used in HealthVault. Review Chapter 4 to make sure you handle these cases.

# Releasing your application to end users

After testing your application thoroughly, including the conditions listed in the previous section, you are ready to release it to end users. The HealthVault team has documented the release process, termed the Go-Live Process, on the HealthVault developer network at <http://msdn.microsoft.com/en-us/healthvault/bb962148>.

The first step in this process it to ensure that you have a signed a business agreement with HealthVault, and have an identifier associated with your partner account. This is a non-technical step and could be done way before your application is ready to be released.

Having established a partner account with Microsoft, you can submit a request to the technical team to review your application in the pre-production environment and push it to the HealthVault production environment. The review typically tests that your application plays well in the HealthVault ecosystem and uses the brand appropriately.

An important step, once your application is available for the world to use, is to network with fellow applications! This can be done using the Wiki provided by the HealthVault team at http://partners.mshealthcommunity.com/hv\_eco/w/wiki/partner-directory.aspx. In addition to business networking, this wiki is used by applications to notify the development community of any extensions to data types that they have implemented. Having well documented extensions available makes it easy for other applications to work with data created by your application, giving you the benefit of network effects.

The last step in your Go-Live Process is to make sure that your application is discoverable to end users. This typically means working with the HealthVault team to become part of their application directory at http://www.healthvault.com.

# Monitoring and maintaining your application

Congratulations on getting that application out there! Whether you have created a client or a web application, it’s very important to monitor its health.

You should log all the failed calls to the HealthVault web service. For additional debugging, the HealthVault SDK provides a tracing mechanism which you can use to log all the request responses. The mechanics of it are detailed at <http://msdn.microsoft.com/en-us/library/ff803626.aspx>. The HealthVault team monitors their development forums available at http://www.msdn.com/healthvault, and you can use them to report any anomalies or failures in the service.

Each release of the HealthVault .NET SDK is supported for two years, and the team frequently adds enhancements and bug fixes to the newer releases. SDKs and libraries available in other language--Java, Python, etc. -- are also updated by the community. As part of maintaining your application, you should make sure you monitor the underlying libraries so that you can upgrade your service to use the most robust offerings.

Another important aspect of maintaining a HealthVault application is to maintain its security artifacts, such as the application certificate and user tokens. HealthVault uses X509 certificates to authenticate web applications; you should make sure that the certificates you use for your application have the appropriate validity to function for a long time. HealthVault uses long-lived user tokens for client applications and you should make sure that these applications frequently refresh the tokens.

# Adding new features to your application

Having a well-tested and well-maintained usable application will frequently result in a number of feature requests from users, not a bad problem to have! Many of these feature requests will necessitate support for additional HealthVault data types.

When you are updating your application to have access to new HealthVault data types, you must request the user to re-authorize your application so that you can get access to these additional data types. Another feature available from HealthVault is optional rules; these rules are data type authorization rules that ask permission for only additional data types. In addition to providing a smooth upgrade curve, optional rules also enable you to run an older version of your application side by side with the new version in case your users prefer not to upgrade. You can read more about optional rules at <http://msdn.microsoft.com/en-us/library/ff803609.aspx>.

Updating the data type rules of an application is not automatic, and an application typically needs to go through the HealthVault Go-Live process at <http://msdn.microsoft.com/en-us/healthvault/bb962148> to release this update in HealthVault production environment.

# Taking your application international!

Throughout the book, we have worked with HealthVault as it is available in United States. However HealthVault has a growing list of implementation partners and the platform is available in Canada, the UK, and Germany as of this writing. You can work with the HealthVault team to explore releasing your application in each of these countries.

The important aspect to keep in mind is that HealthVault as a service is completely globalized and internationalized. The request and response string HealthVault displays to a user can be changed to appropriate locales. In fact, you can also make a Spanish version of your application available in the United States. The HealthVault application configuration center through its *Localize* tab, allows developers to configure their applications with Spanish strings so that when a user with Spanish browser settings accesses HealthVault, the application’s authorization screen is shown with the appropriate language. The HealthVault Shell redirect interface, which is discussed in detail in Chapter 3, also respects an ***lcid***= {*Locale ID*} parameter in its query string to show the appropriate display language in the user interface.

Additionally, HealthVault offers an internationalized set of units and time formats to allow applications to work with appropriate standards in the target country.

With this information, you are all set to stride into the exciting world of enabling quantified self with HealthVault!

# Further Resources

This section highlights some important resource available for HealthVault development.

## Need reference information?

The HealthVault MSDN Site, http://www.msdn.com/healthvault, is a great resource on all things HealthVault. HealthVault features and SDK information is available in the reference section at http://msdn.microsoft.com/en-us/library/aa155110.aspx. The HealthVault team has an active blog at http://blogs.msdn.com/b/healthvault/ and a list of frequently asked questions at http://blogs.msdn.com/b/healthvaultfaq/.

## Have a Question?

The HealthVault Forums at http://social.msdn.microsoft.com/forums/en-US/healthvault/ are a great place to ask technical questions.

## Development tools

Fiddler is a great tool to enable request – response tracing for web applications. This tool will help you look at and analyses XML information being exchanged with HealthVault platform.

Get Real Consulting’s X-Ray for HealthVault at https://xray.getrealconsulting.com/ is an indispensable tool for HealthVault development. It offers the ability to create data in a HealthVault pre-production and production environment and export & import information.

The HealthVault team hosts the developer center tool at http://developer.healthvault.com/; it is very handy to look at HealthVault method and data type schemas.

HealthVault Data Type Checkup at http://datacheckup.healthvault-ppe.com/hvappcheckup, offers lint functionality for data written by your application. This tool highlights best practices for writing data to HealthVault for a select set of data types.

## Mapping your data to HealthVault

Chapter 4 summarizes the intent and use of various data types in HealthVault. Didier Thizy has a good post at http://www.macadamian.com/insight/healthcare\_detail/mapping\_hl7\_phm\_to\_healthvault/ on mapping the HL7 PHM standard to HealthVault.

The HealthVault team maintains a detailed mapping of the ASTM Continuity of Care Record standard to HealthVault data types on MSDN at <http://msdn.microsoft.com/en-us/healthvault/ee663895>. The reference article at <http://msdn.microsoft.com/en-us/library/ff803579.aspx> is a great resource for using CCR data in HealthVault.